Memorandum 1971 (December 2011). ISSN 1874-4850. Available from: http://www.math.utwente.nl/publications
Department of Applied Mathematics, University of Twente, Enschede, The Netherlands

A Note on Perfect Partial Elimination

Matthijs Bomhoff*, Walter Kern†, and Georg Still‡

University of Twente, Faculty of Electrical Engineering, Mathematics and Computer Science, P.O. Box 217, 7500 AE Enschede, the Netherlands

December 31, 2011

Abstract

In Gaussian elimination it is often desirable to preserve existing zeros (sparsity). This is closely related to perfect elimination schemes on graphs. Such schemes can be found in polynomial time. Gaussian elimination uses a pivot for each column, so opportunities for preserving sparsity can be missed. In this paper we consider a more flexible process that selects a pivot for each nonzero to be eliminated and show that recognizing matrices that allow such perfect partial elimination schemes is NP-hard.

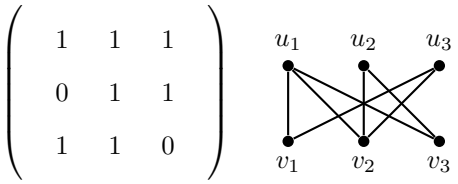
keywords Gaussian elimination, perfect elimination, bipartite graph, complexity

1 Introduction

Sparse matrices commonly occur in practical applications. When performing Gaussian elimination on such matrices it is often desirable to preserve sparsity by avoiding *fill-in* – the process of turning a zero element into a nonzero. Avoiding fill-in completely during elimination, so-called *perfect elimination*, has been treated extensively in literature, both for the general case of square matrices [1, 2, 3] and for special cases such as symmetric matrices or pivots on the diagonal [4, 5]. For each of these cases, determining whether perfect elimination is possible can be done in polynomial time.

It is characteristic for the Gaussian elimination algorithm that in each iteration a pivot element is picked and used to clear its entire column. If we want to avoid fill-in completely, this limits the set of matrices we can apply this method to. In order to achieve perfect elimination for a broader set of matrices, we can use a more fine-grained elimination process in which we eliminate single nonzero values instead of entire columns at a time. Rose and Tarjan already mentioned such *partial elimination* as an alternative to ordinary Gaussian elimination for this reason [5]. In this paper we show that determining whether a matrix allows

*m.j.bomhoff@utwente.nl
†w.kern@utwente.nl
‡g.j.still@utwente.nl



(a) (b)

Figure 1: A $\{0,1\}$-matrix M and its bipartite graph $G[M]$.

perfect partial elimination is NP-hard. The remainder of this paper is organized as follows: The next section introduces the $\{0,1\}$-matrix and bipartite graph representations of the problem as well as the concept of perfect elimination. The third section formalizes the more fine-grained proces we analyze. The section after that contains our main result on the NP-hardness of the related recognition problem. Finally, the fifth section contains some concluding remarks.

2 Perfect Elimination

Our main interest in this paper is recognizing those matrices that admit perfect elimination schemes. Under the assumption that subtracting a multiple of one row from another will not lead to 'accidental' cancellations besides zeroing the intended element, we can represent an instance of this problem by a $\{0,1\}$-matrix M containing zeros in exactly the same places as the original matrix. In the $\{0,1\}$-matrix representation, subtraction takes a different form: When subtracting two rows, only a single 1 is turned into a zero. This represents the assumption that no additional cancellations outside of the pivot column occur. Interpreting this matrix as a biadjacency matrix leads to a bipartite graph $G[M] = (U, V, E)$ with the rows and columns of M as its vertex classes U and V. An example matrix M and its associated bipartite graph $G[M]$ are shown in Fig. 1. Pivots suitable for perfect Gaussian elimination on M correspond to so-called *bisimplicial* edges of $G[M]$:

Definition 2.1. An edge uv of a bipartite graph $G = (U, V, E)$ is called *bisimplicial* if the neighbors of its endpoints $\Gamma(u) \cup \Gamma(v)$ (where $\Gamma(u)$ denotes the neighbors of u) induce a complete bipartite subgraph of G.

Using this definition, the perfect elimination problem for bipartite graphs was first defined by Golumbic and Goss [1] as follows:

Definition 2.2. A bipartite graph $G = (U, V, E)$ is called *perfect elimination bipartite* if there exists a sequence of pairwise nonadjacent edges $[u_1v_1, \ldots, u_nv_n]$ such that u_iv_i is a bisimplicial edge of $G - \{u_1, v_1, \ldots, u_{i-1}, v_{i-1}\}$ for each i and $G - \{u_1, v_1, \ldots, u_n, v_n\}$ is empty. Such a sequence of edges is called a *(perfect elimination) scheme*.

The recognition of perfect elimination bipartite graphs corresponding to matrices that allow Gaussian elimination without fill-in is possible in polynomial time and several algorithms for this have been published [1, 2, 3].

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

(a) (b)

Figure 2: Two matrices, both without a perfect elimination scheme, but (a) has a perfect partial elimination scheme whereas (b) does not.

$$\begin{pmatrix} \boxed{1} & \boxed{1} & 0 & 0 \\ \boxed{1} & \boxed{1} & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

Figure 3: The disposable elements (*indicated by squares*) of our example matrix.

3 Perfect Partial Elimination

In Gaussian elimination a single pivot element is used to zero its entire column. When trying to avoid fill-in, this process can be too restrictive and it may be beneficial to perform partial pivots, i.e., to select a new pivot element for every single nonzero element that we zero. For example, the matrix in Fig. 2(a) can be diagonalized by partial pivots without fill-in – although no perfect elimination scheme exists. Clearly, there are also matrices like the one in Fig. 2(b) for which even partial pivoting cannot avoid fill-in. It is thus natural to ask ourselves which matrices allow perfect elimination using such partial pivots.

To answer this question, reconsider the elimination of edges in the corresponding bipartite graph. As the partial elimination steps involve only row operations zeroing single nonzero elements in our matrix, we first look at elements that can be zeroed this way. Recall that U denotes the vertex class corresponding to the rows of the matrix and V denotes the vertex class corresponding to the columns of the matrix.

Definition 3.1. An edge uv of a bipartite graph $G = (U, V, E)$ is called *disposable* if there exists another edge $u'v$ such that $\Gamma(u') \subseteq \Gamma(u)$.

Before defining the analogous concept for a $\{0,1\}$-matrix M, we first introduce the $\leq$ relation on matrix rows. We write $M_{i',*} \leq M_{i,*}$ to denote that for any column j of M, $M_{i',j} = 1$ implies $M_{i,j} = 1$. Note how this captures the same notion expressed by $\Gamma(u') \subseteq \Gamma(u)$ in the bipartite graph case. Defining disposable elements in the $\{0,1\}$-matrix M is now rather straight-forward: a nonzero element $M_{i,j}$ of a row $M_{i,*}$ is called *disposable* if there is another row $M_{i',*} \leq M_{i,*}$ such that $M_{i',j}$ is also nonzero. (Element $M_{i',j}$ can be used as a

$$\begin{pmatrix} 10 & 1 & 0 & 0 \\ 2 & 9 & 0 & 0 \\ 3 & 0 & 8 & 5 \\ 0 & 4 & 6 & 7 \end{pmatrix}$$

Figure 4: An example perfect partial elimination scheme.

partial pivot to clear element $M_{i,j}$.) Fig. 3 shows the disposable elements of our example matrix. Clearly, disposable elements play an important role in the characterization of the matrices that allow perfect partial elimination schemes. We now come to the definition of the class of bipartite graphs associated to these matrices:

Definition 3.2. A bipartite graph $G = (U, V, E)$ with $|U| \geq |V| = n$ and $|E| = m$ is called *perfect partial elimination bipartite* if there exists a bijection $f : E \to \{1, \ldots, m\}$ (denoted by $E = \{e_1, \ldots, e_m\}$) such that $\{e_{m-n+1}, \ldots, e_m\}$ together form a maximum matching of G covering V and for each $i \in \{1, \ldots, m-n\}$, e_i is a disposable edge of $G_i := (U, V, \{e \in E \mid f(e) \geq i\})$. We call the bijection f a *perfect partial elimination scheme*.

As our partial pivots only involve single edge operations, it is no longer required for the two vertex classes to be of equal size. Clearly, the notion of a perfect partial elimination scheme for G carries over readily to the corresponding $\{0,1\}$-matrix, which also no longer has to be square. An example of a perfect partial elimination scheme for the $\{0,1\}$-matrix of Fig. 2(a) is shown in Fig. 4.

Having defined the class of matrices and bipartite graphs that allow perfect partial elimination, the logical next question to ask is how hard it is to recognize members of this class. The Perfect Partial Elimination decision problem regarding this can be stated as follows:

Instance: A $\{0,1\}$-matrix M

Question: Does M have a perfect partial elimination scheme?

Our main result states that Perfect Partial Elimination is NP-complete.

4 Main Result

The proof is by reduction from Satisfiability [6]. We start by briefly defining this problem, using the terminology and notation from Garey and Johnson [7]. An instance S of Satisfiability consists of a set U of Boolean variables and a set C of clauses. For every variable $u_i \in U$, u_i and $\bar{u}_i$ are called *literals* over U. A *truth assignment* (T, F) is a partition of the variables U. Under a given truth assignment, the literal u_i is true if and only if $u_i \in T$, otherwise it is false. Similarly, the literal $\bar{u}_i$ is true if and only if $u_i \in F$. The clauses in C are disjunctions of the literals over U. A given truth assignment is called *satisfying* for C if every clause in C contains at least one literal that is true under the truth assignment. The Satisfiability decision problem is now stated as follows:

Instance: Set U of variables, collection C of clauses over U.

Question: Is there a satisfying truth assignment for C?

We can now prove our main result.

Theorem 4.1. Perfect Partial Elimination *is NP-complete.*

Proof. As a given elimination sequence for perfect partial elimination can clearly be verified in polynomial time, we only show NP-hardness using a reduction from Satisfiability. For a given instance $S = (U, C)$ of Satisfiability, we construct a corresponding $\{0, 1\}$-matrix M_S such that M_S has a perfect partial elimination scheme if and only if S has a truth assignment for U that is satisfying for C. To simplify the reduction, we will assume w.l.o.g. that the instance S has at least one clause, all clauses contain at least one literal, and C contains no tautologies, i.e., there is no i such that some clause contains both u_i and $\bar{u}_i$.

The matrix M_S has $4|U| + |C| + 1$ rows and $2|U| + |C| + 2$ columns. For the description of the construction procedure, it is convenient to label the rows and columns instead of referring to them simply by number.

For every variable u_i in U we have two columns labeled u_i and $\bar{u}_i$. These columns are used to represent the variables, their truth assignments and their occurrences in the clauses. For every clause $c_i \in C$, we have a column labeled c_i. These columns are used to keep the individual clauses separated while linking them together in the overall satisfiability requirement. We also have two auxiliary columns labeled a and b, which are used to limit the possible subtractions between rows.

The rows of M_S are partitioned into five sets: the first two sets each contain one row per variable and are denoted V and W. Subtractions between rows of these sets are used to represent possible truth assignments for U. The third set D is used mainly to clear matrix elements that are no longer required for the elimination process themselves. The fourth set of rows, K, represents the clauses of S and links them to their literals. The final set R contains only a single row and encodes the requirement that all clauses must be satisfied by the truth assignment.

Having introduced the rows and columns that together form the constructed matrix M_S, we will now describe the values of the elements of M_S. The row set V contains a single row v_i for every variable u_i. Each such row contains two ones in the columns corresponding to u_i and $\bar{u}_i$ and zeros everywhere else. The rows w_i in W are identical to the rows v_i, except for an additional 1-entry in column a. The set D contains two rows for each variable u_i: one for each of the two corresponding literals u_i and $\bar{u}_i$. Each row in D has a one in the corresponding literal column and a one in column b and zeros everywhere else. The rows in K each correspond to a clause in C. Row k_i has a one in every column corresponding to a literal occurring in c_i, as well as a one in the column corresponding to c_i itself. All rows in K also have a one in the columns a and b and zeros elsewhere. Finally, the set R contains only a single row r with ones in all columns c_i as well as in the b column, and zeros everywhere else. An example of this construction for $S = (U, C)$ with $U = \{u_1, u_2, u_3, u_4\}$ and $C = \{\{u_1, u_2\}, \{u_1, \bar{u}_2, u_3, \bar{u}_4\}, \{u_1, \bar{u}_3, u_4\}\}$ is shown in Fig. 5 where the nonzero entries have been numbered according to the elimination scheme that will be described next. To complete the reduction, we have to show that M_S has

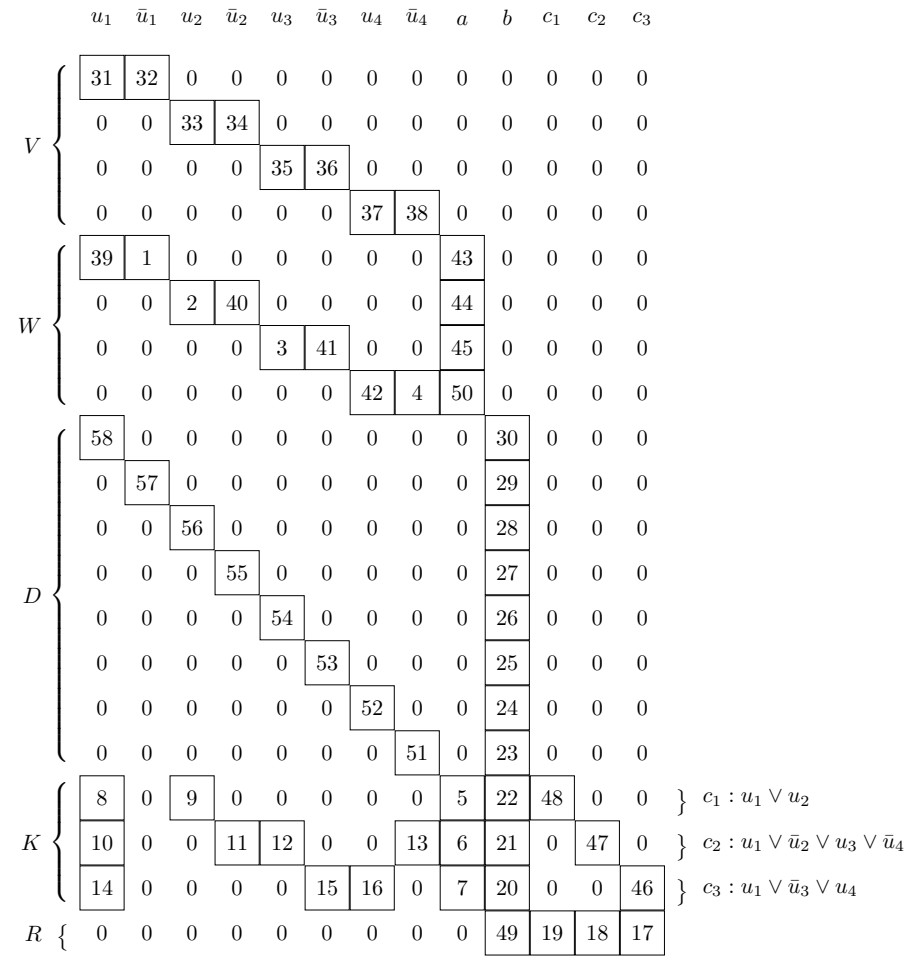


Figure 5: An example perfect partial elimination scheme for the construction used in the NP-hardness proof of Perfect Partial Elimination (*nonzero entries emphasized by squares*).

a perfect partial elimination scheme if and only if S is satisfiable. We first show how a satisfying truth assignment for S leads to a perfect partial elimination scheme for M_S.

Let (T, F) be a satisfying truth assignment for S. For every $u_i \in T$, we use the corresponding row in V to clear the element in the $\bar{u}_i$ column of the corresponding row in W. Similarly, for every $u_i \in F$, we use the corresponding row in V to clear the element in the u_i column of the corresponding row in W. The modified rows in W now represent the truth assignment (T, F). As (T, F) is a satisfying truth assignment for S, we can find a row w_j for each clause row k_k such that $w_j \leq k_k$. We use these rows to clear all elements in the a column of K. Next, the rows from D are used to clear all the elements in the u_i and $\bar{u}_i$ columns of K. The only nonzero elements remaining in K are now the column b and the diagonal in the c_i columns. For every clause c_i we now have that $k_i \leq r$. The rows of K are now used to clear all c_i columns of r, so that the

only nonzero element of r that remains is in column b. The remainder of the elimination scheme is rather straightforward: The current row r can be used to clear column b in row sets D and K. The resulting rows in D can then zero all of the literal entries in other rows, leading to a matrix in which for each column there is a row with only a single 1-entry in exactly that column. After that, completion of the elimination scheme is a trivial task. Figure 5 shows an example of such an elimination scheme.

It remains to show that no perfect partial elimination scheme exists if S is not satisfiable. We start with two key observations: First of all, only rows of K can be used to clear the nonzero elements in the c_i columns of r. At least $|C| - 1$ such operations using different rows of K need to be performed before row r itself can be used to clear an element of another row. Secondly, in order to use rows of K to clear elements in r, we need to first clear their nonzero elements in the a column. The only rows with a nonzero element in this column that could be used to accomplish this are the rows of W. Clearly, using only rows in V, W and D, we cannot create a row with only a nonzero element in the a column, so in particular each of the rows in W will have at least one literal column with a nonzero element. Furthermore, as we excluded tautologies from our clauses, a row w_i can only be used as a pivot after clearing exactly one of its two nonzero values in the literal columns (using the corresponding row v_i). However, performing this process on all rows of W again represents a truth assignment for S. Therefore no perfect partial elimination scheme can exist if at most $|C| - 2$ clauses can be satisfied by any truth assignment.

If there does exist a truth assignment that satisfies all but one clause, it can be used to eliminate all but one nonzero value in the c_i columns of r. After that, row r contains two nonzeros: one in the c_i column corresponding to the clause that is not satisfied, and one in the b column. Row r can then in turn be used to clear the nonzero value in either the c_i column or the b column of the row corresponding to the non-satisfied clause. Either way, this row corresponding to the non-satisfied clause will retain at least two nonzero elements among the c_i, a and b columns, blocking the remainder of the elimination process, as no other row remaining at this point has at least two nonzero elements among these columns. Thus no perfect partial elimination scheme can exist in this case either. □

Remark 4.2. The problem does not become easier if we restrict ourselves to square matrices. Indeed, we can augment the matrix M_S with additional 'all ones' columns. A moment of reflection shows that these additional columns neither prevent a perfect partial elimination scheme if S has a satisfying truth assignment, nor do they allow such a scheme if S does not have a satisfying truth assignment.

5 Conclusion

When performing Gaussian elimination on sparse matrices, the choice of pivots is critical to preserving sparsity. Ideally, during elimination not a single zero element is turned into a nonzero. Previous work on the relation between Gaussian elimination and elimination schemes on bipartite graphs has led to recognition algorithms for the class of matrices and graphs that allow such per-

fect elimination schemes. However, by being restricted to a single pivot per column, some possibilities for preserving sparsity may be missed. By clearing single elements at a time instead of performing pivots on entire columns at once, a more fine-grained variant on Gaussian elimination can achieve perfect elimination on a larger class of matrices and their associated bipartite graphs. However, our main result shows that determining whether a matrix allows such a perfect partial elimination scheme is NP-hard.

As our analysis only treats the general case, it may be interesting to also investigate whether more restricted classes of matrices do admit a polynomial time algorithm for partial elimination. Another subject for possible further research could be parameterized versions of the Perfect Partial Elimination decision problem, for example bounding the degree of vertices in the bipartite graph.

Acknowledgement.

We gratefully acknowledge the support of the Innovation-Oriented Research Programme 'Integral Product Creation and Realization (IOP IPCR)' of the Netherlands Ministry of Economic Affairs, Agriculture and Innovation.

References

[1] M. C. Golumbic, C. F. Goss, Perfect elimination and chordal bipartite graphs, J. Graph Theory 2 (2) (1978) 155–163.

[2] L. Goh, D. Rotem, Recognition of perfect elimination bipartite graphs, Inform. Process. Lett. 15 (4) (1982) 179–182.

[3] J. P. Spinrad, Recognizing quasi-triangulated graphs, Discrete Appl. Math. 138 (1-2) (2004) 203–213.

[4] D. J. Rose, R. E. Tarjan, G. S. Lueker, Algorithmic aspects of vertex elimination on graphs, SIAM J. Computing 5 (2) (1976) 266–283.

[5] D. J. Rose, R. E. Tarjan, Algorithmic aspects of vertex elimination on directed graphs, SIAM Journal on Applied Mathematics 34 (1) (1978) 176–197.

[6] S. A. Cook, The complexity of theorem-proving procedures, in: Proceedings of the third annual ACM symposium on Theory of computing, STOC '71, ACM, New York, NY, USA, 1971, pp. 151–158.

[7] M. R. Garey, D. S. Johnson, Computers and Intractability, W.H. Freeman and Company, San Francisco, 1979.